Machine Poetize Using DeepLearning Models

Yidan Tian
Department of Statistics
The University of Michigan
Ann Arbor, USA
ydantian@umich.edu

***Abstract*—This study explores classical Chinese poetry generation using deep learning models. Logistic Regression, RNN, and LSTM were implemented and evaluated on the Chinese Poems dataset. Results highlight the strengths and limitations of each model in terms of accuracy, computational efficiency, and learning capacity. Future improvements focus on addressing gradient issues and optimizing model hyperparameters to enhance performance.**

I. Introduction

A. Background and Motivation

Poetry generation is a complex and culturally rich task that combines artistic creativity with artificial intelligence challenges. Classical Chinese poetry, in particular, is characterized by strict structural rules such as tonal balance, semantic coherence, and rhythmic flow. Recent advancements in machine learning, especially deep learning, have enabled machines to generate human-like text, opening new frontiers for machine creativity.

This project explores poetry generation through a range of models, from basic statistical approaches to sophisticated deep learning architectures. The motivation of this work lies in understanding the strengths and limitations of these models for Chinese poetry generation.

B. Project Goal

The goals of this project are as follows:

- Develop and implement distinct models for poetry generation: Logistic Regression, RNN, and LSTM.
- Compare model performance on metrics such as semantic coherence, structural adherence, and computational efficiency.
- Analyze how different architectures learn and replicate the stylistic intricacies of Chinese classical poetry.

C. Literature Review

Recent advancements in deep learning have significantly improved poetry generation, addressing challenges such as semantic coherence and long-term dependencies. Zhang et al. [1] provide a foundational review of Chinese poetry generation, emphasizing the effectiveness of RNNs, LSTMs, and Transformer-based models. Song et al. [2] introduce controllable generation methods that maintain precise stylistic features, such as tone and rhythm. Vaswani et al. [3] revolutionized sequence modeling with the Transformer architecture, enabling efficient handling of long-term dependencies, while Tian and Li [4] demonstrate the superiority of Transformer-XL for Chinese poetry by improving structural accuracy and semantic relevance. Zhu et al. [5] highlight the emotional expressiveness achieved through hybrid models that combine RNNs and attention mechanisms, and Wu et al. [6] propose noise-aware training strategies to enhance the robustness and quality of generative outputs. Together, these works provide critical insights into model design and training strategies, which inform the comparative analysis of RNNs, LSTMs, and ERNIE-GEN in this project.

II. Method

A. Problem Formulation

The project aims to generate classical Chinese poetry using machine learning, framed as a sequence generation task. It employs the Chinese Poems dataset from HuggingFace, with over 218,000 poems. Preprocessing includes word-level tokenization, fixed-length padding, and excluding poems over 100 characters. Tokenized inputs are converted into word embeddings, enabling models to learn patterns and generate coherent poetry.

B. Methodologies

The methodologies include a data preprocessing pipeline, where raw poems are cleaned, tokenized, and converted into uniform sequences. Padding is applied to standardize sequence lengths, and word embeddings are employed to transform tokens into meaningful vector representations. Logistic regression is first implemented as a baseline model to predict the next word based on the preceding three words. The RNN and LSTM models are then developed using PyTorch or TensorFlow frameworks. Both models take padded input sequences and output predictions for the next word, with cross-entropy loss used as the training criterion. The RNN processes the input sequentially while maintaining a single hidden state, whereas the LSTM introduces both short-term and long-term memory states, enabling it to better retain context over extended word sequences.

C. Tools and Frameworks

Python serves as the core programming language, with libraries such as NumPy and Pandas used for data preprocessing and analysis. The PyTorch and TensorFlow frameworks are employed for developing and training the RNN and LSTM models. Visualization tools, such as Matplotlib and Seaborn, are used to analyze training loss, model convergence, and performance comparisons.

III. Results

The models were trained on NVIDIA A100 80GB PCIe.

A. Logistic Regression Model

TABLE I
LOGISTIC REGRESSION MODEL SUMMARY

Layer (Type)	Output Shape	Param #
Input (InputLayer)	(None, 3)	0
Embedding (Embedding)	(None, 3, 256)	727,808
Flatten (Flatten)	(None, 768)	0
Dense (Dense)	(None, 2843)	2,186,267
Total Parameters		**2,914,075**

The Logistic Regression model served as a baseline and achieved a high training loss with poor accuracy. It lacks the ability to retain sequential dependencies, which limits its performance for poetry generation. The model converged slowly, showing minimal improvement over epochs.

B. RNN Model

TABLE II
RNN MODEL SUMMARY

Layer (Type)	Output Shape	Param #
Input (InputLayer)	(None, 48)	0
Embedding (Embedding)	(None, 48, 64)	181,952
SimpleRNN (SimpleRNN)	(None, 48, 128)	24,704
Dense (Dense)	(None, 48, 2843)	366,747
Total Parameters		**573,403**

The RNN model is efficient and lightweight compared to Logistic Regression. The RNN successfully captured short-term dependencies within poetic sequences; however, its loss curve indicates saturation after three epochs, suggesting limitations in handling long-term dependencies.

C. LSTM Model

TABLE III
LSTM MODEL SUMMARY

Layer (Type)	Output Shape	Param #
Input (InputLayer)	(None, 49)	0
Embedding (Embedding)	(None, 49, 128)	363,904
LSTM (LSTM)	(None, 49, 128)	131,584
Dropout (Dropout)	(None, 49, 128)	0
Bidirectional (Bidirectional)	(None, 49, 128)	98,816
Dense (Dense)	(None, 49, 2843)	366,747
Total Parameters		**961,051**

The LSTM model includes an LSTM layer followed by a dropout layer and a bidirectional LSTM. This configuration increases the total parameters to 961,051, making it more complex but potentially better at capturing long-term dependencies. The LSTM model was introduced to address the limitations of the RNN. LSTM's architecture, which includes mechanisms to retain both short-term and long-term memory, resulted in a stable training process. However, despite its sophisticated design, the model displayed a surprising accuracy plateau around 45.59%, likely due to hyperparameter tuning challenges and potential issues with gradient vanishing.

D. Training and Validation Trends

The RNN model showed fast convergence with minor overfitting towards the later epochs, as the validation accuracy closely matched training accuracy. The LSTM model, on the other hand, displayed a high initial accuracy that failed to improve significantly due to NaN values observed in the training process. This suggests instability in learning, possibly arising from an aggressive learning rate or exploding gradients.

IV. Conclusion

This study compared three deep learning models for classical Chinese poetry generation: Logistic Regression, RNN, and LSTM. Key findings include:

- RNNs are effective for modeling short-term dependencies but are limited by vanishing gradients for longer sequences.
- LSTMs provide theoretical advantages for long-term dependencies but require careful hyperparameter tuning and gradient management to avoid instability.
- Logistic Regression, while computationally lightweight, is unsuitable for complex sequential tasks like poetry generation.

To further improve the LSTM model, the following strategies can be explored:

1) Adjust the learning rate schedule to ensure smoother convergence.
2) Use pre-trained embeddings such as ERNIE-GEN to provide semantic richness to the input data.
3) Fine-tune the model with a smaller batch size to enhance stability during training.

References

[1] Z. Zhang, H. Luo, and J. Chen, A Comprehensive Review of Chinese Poetry Generation Using Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 421–435, 2023.

[2] Y. Song, C. Zhang, and K. Xu, Controllable Poetry Generation via Multi-Granularity Structures," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 1, pp. 2234–2242, 2023.

[3] A. Vaswani et al., Attention Is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[4] Y. Tian and X. Li, Chinese Poetry Generation Using Transformer-XL and Semantic Embeddings," *Journal of Computational Linguistics*, vol. 49, no. 1, pp. 67–85, 2023.

[5] J. Zhu, M. Huang, and L. Zhang, DeepPoetry: A Hybrid Model for Generating Emotional Poetry," *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1421–1432, 2022.

[6] K. Wu, Y. Zhang, and S. Zhou, Exploring Noise-Aware Training in Generative Language Models for Classical Poetry," *IEEE Access*, vol. 11, pp. 31922–31935, 2023.